UniClever

ENGIE4800 - Data Science Capstone

Ian Johnson (icj2103@columbia.edu) & Julien Maudet (jm4418@columbia.edu)
GitHub: https://github.com/Unilever-NLP/unilever-nlp

May 8th, 2017

Contents

1. Introduction

In this project, which is a collaboration between Columbia University and the Consumer Science Department at Unilever, we developed UniClever, a web interface that enables analysts at Unilever to extract meaningful information from textual survey data that they gather from customers around the globe, or from Amazon reviews of their beauty products.

The web interface performs two main tasks: a summarization of all answers to a given question or of all reviews for a given product and the extraction of the most relevant keyphrases in the answers.

In this report, we will start with introducing the datasets then move on to presenting both tasks and finally the web interface.

2. Description of the Datasets

We work with two types of datasets here, amazon reviews on beauty products and survey data.

2.1 Survey data

It consists in 198, 468 survey answers to the following questions:

- What is healthy skin?
- How do you know your skin is healthy?
- How do you know your skin is getting healthier with every shower?
- How do you get healthy skin?
- How do you maintain healthy skin?
- How bar soap or body wash gives you healthy skin?
- How concerned?

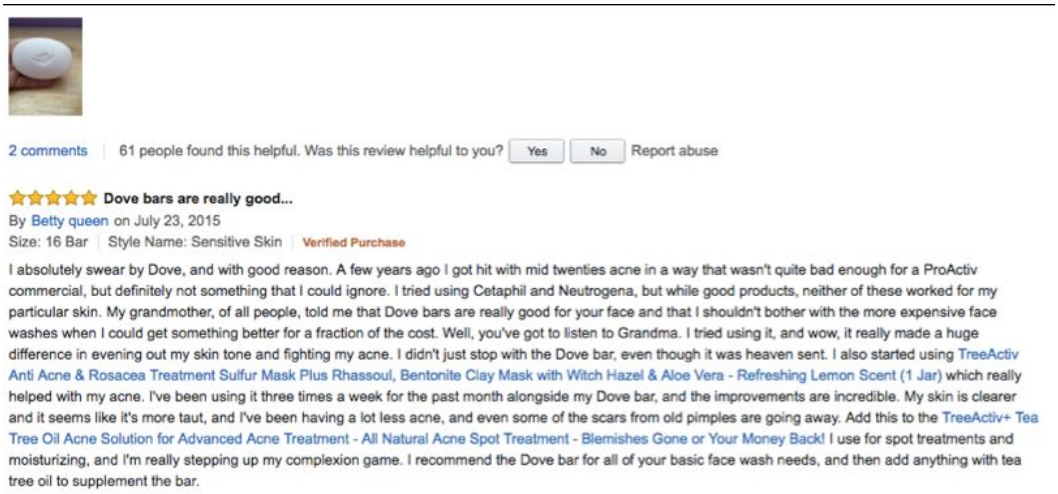
Here's a snapshot of the dataset:

What is Healthy Skin?	How do you know your skin is healthy?	How do you know your skin is getting healthier with every shower?
smooth and soft	NO SPOTS OR LARGE PORES	i dont
Skin free of blemishes	Because it is free of blemishes	My pores care closing
no irritations	on irritations or rashes	not dry
Skin that is not dry, is soft to the touch, not a lot of cracks or wrinkles.	It has a slight shine to the surface	I have no idea
Skin that feels soft and has a nice glow to it	If it looks like it flows without any make up	It feels softer and tighter
Glowing even skin tone	No dry spots or blemishes	Not dry and flaky
soft, moisturized, non-blemished	not dry, not blemished, healthy color	don't know
Skin that is not dry, has a golden glow, and is not wrinkled or leathery.	I can feel it and see it's appearance. If it is off, then I know there is something wrong and I need to fix it.	I can see suttle improvements.
Skin that is not dry and seemingly glows	If it is firm and glowing with out a whitish tint from dryness	I don't know
Skin that is moisturized and clean.	Texture and color	Feeling refreshed and vibrant
Free from skin disorders, soft and clear with minimal wrinkling	how it feels to the touch as well as how it looks	As long as it doesn't feel dry I feel it's healthy

We process the data by concatenating all answers to a given question into a single text. It basically gives us one large character chain per question that we can perform NLP analysis on.

2.2 Amazon reviews

The whole dataset contains 198,502 Amazon reviews that originally look like the following:



2 comments | 61 people found this helpful. Was this review helpful to you? Yes No Report abuse

Dove bars are really good...

By Betty queen on July 23, 2015

Size: 16 Bar | Style Name: Sensitive Skin | Verified Purchase

I absolutely swear by Dove, and with good reason. A few years ago I got hit with mid twenties acne in a way that wasn't quite bad enough for a ProActiv commercial, but definitely not something that I could ignore. I tried using Cetaphil and Neutrogena, but while good products, neither of these worked for my particular skin. My grandmother, of all people, told me that Dove bars are really good for your face and that I shouldn't bother with the more expensive face washes when I could get something better for a fraction of the cost. Well, you've got to listen to Grandma. I tried using it, and wow, it really made a huge difference in evening out my skin tone and fighting my acne. I didn't just stop with the Dove bar, even though it was heaven sent. I also started using TreeActiv Anti Acne & Rosacea Treatment Sulfur Mask Plus Rhassoul, Bentonite Clay Mask with Witch Hazel & Aloe Vera - Refreshing Lemon Scent (1 Jar) which really helped with my acne. I've been using it three times a week for the past month alongside my Dove bar, and the improvements are incredible. My skin is clearer and it seems like it's more taut, and I've been having a lot less acne, and even some of the scars from old pimples are going away. Add this to the TreeActiv+ Tea Tree Oil Acne Solution for Advanced Acne Treatment - All Natural Acne Spot Treatment - Blemishes Gone or Your Money Back! I use for spot treatments and moisturizing, and I'm really stepping up my complexion game. I recommend the Dove bar for all of your basic face wash needs, and then add anything with tea tree oil to supplement the bar.

They are already preprocessed and here is the format of the dataset as we will use it:

	order	reviewrID	asin	reviewerName	helpful	out of	reviewText"	overall	summary	unixReviewTime	reviewTime
0	1	A1YJEY40YUW4SE	7806397051	Andrea	3	4.0	Very oily and creamy. Not at all what I expect...	1	Don't waste your money	1.391040e+09	01 30,2014
1	2	A60XNB876KYML	7806397051	Jessica H.	1	1.0	This palette was a decent price and I was look...	3	OK Palette!	1.397779e+09	04 18,2014
2	3	A3G6XNM240RMWA	7806397051	Karen	0	1.0	The texture of this concealer pallet is fantas...	4	great quality	1.378426e+09	09 6,2013
3	4	A1PQFP6SAJ6D80	7806397051	Norah	2	2.0	I really can't tell what exactly this thing is...	2	Do not work on my face	1.386461e+09	12 8,2013
4	5	A38FVHZTNQ271F	7806397051	Nova Amor	0	0.0	It was a little smaller than I expected,but th...	3	It's okay.	1.382141e+09	10 19,2013

The group-by functionality of the interface handles this dataset specifically by performing analyses on the "reviewText" column, while grouping on the "asin" column which corresponds to separated product identifiers.

3. Summarization

In this section we present our initial work towards implementing the Semantic Volume Maximization [3] algorithm for text summarization on these datasets. The choice to include the summarization functionality into our application was made because of the usefulness that it provided to the Unilever datasets, as compared to other textual data manipulations and exploratory analyses. Other methods that we considered included clustering of similar documents, review scores prediction, LDA, and sentiment analysis. As compared to these alternatives, the summarization and keywords ranking stood out as the most useful to the task of providing insight into the datasets, and were also amenable to inclusion in an application interface. Keywords extraction and extractive summarization seemed to be a useful combination of different methods, as the former provides numerical estimates of the top concepts exhibited by the datasets, and the latter aims to more broadly cover all of the existing information outside of these top keywords.

Our choice of summarization algorithm came from previous research into summarization methods by Safyan et al. (2016), who found that the methods used by Yogotama et al. (2015) performed well on the Unilever datasets, and that Semantic Volume Maximization was both fast flexible in terms of input. In addition, this algorithm was thought to be useful for the current task of summarization across a wide range of data sources (e.g. many different reviewers or survey respondents), who might use a diversity of vocabulary to describe these products and answer these questions.

3.1 Preprocessing

The first step of the summarizer pipeline is to make sentences from reading in the dataframe using the Nltk sentence tokenizer. After this step, we include the option to split longer sentences with a parameter designating the number of words at which to split. This is a preliminary step towards solving the issue of longer sentences being more likely to appear in the final summary, as their distance from the centroid is larger, and thus makes it more likely for these sentences to have a larger distance from their projections onto the basis vector subspace. We might benefit from exploring the notion of normalizing all sentence vectors to unit length beforehands, to avoid this issue. In addition, we will explore the idea of splitting sentences based on conjunctions and commas or other types of delimiters, to keep sentences to within a certain length of one another.

At this point in the pipeline, we had previously attempted to spell check each sentence and correct grammatical and spelling errors using the TextBlob package, but

this tool was too slow to be run against a large corpus, and created a bottleneck, so we have to to be able to integrate this feature, and may explore other spelling correction options to increase readability of user-generated data.

The next pipeline step is to stem or lemmatize each of the sentences. These methods served the purpose of cutting down words so that they would be considered the same when they had different endings. For instance if a sentence had the word “smooth” and another sentence had the word “smoother” then these would be chopped down to just “smooth” and we would be less likely to choose both of these sentences in our final set, as they would be detected as both representing more similar concepts. At first stemming seemed like the best method for this type of normalization, however it has a tendency to chop unrelated words to the same stem, which would be incorrectly assigning a similar vector to two sentences with previously dissimilar words. Lemmatization on the other hand uses a vocabulary and morphological analysis of the words to more accurately convert the words to their proper lemma, as opposed to simply chopping off the last characters. This provides a more suitable method for maintaining the correct distances between the basis vectors.

The next step used in Yogotama et al. is to remove all bigrams from the vector representations that contain only stopwords. This is a cleaning step that is meant to reduce the noise in the vector representations that is due to words of little relevance to our topics.

The next step in preprocessing is to perform the vectorization of the sentences. Yogotama et al. perform this vectorization using bigram counts (e.g. each dimension in the vector represents the number of occurrences of that particular pair of words in the sentence). This bigram count representation can be achieved using the FeatureHasher from Scikit Learn. We also include the option to perform vectorization based solely on count of individual words, using Scikit Learn’s CountVectorizer. We use scipy sparse matrices to preserve space and for efficient computations in later steps.

3.3 Singular Value Decomposition

The final, and most interesting step before feeding the vectors and sentences to the summarization algorithm involves the use of the SVD to represent the term counts in a more condensed form. SVD is used in latent semantic analysis and latent semantic indexing as a method to extract out representations of “topics” from our vectorized documents. These topics can be thought of as groups of words that are often found together in the same documents. This method does not give us a title for these related words - that we have to interpret for ourselves. For example, one topic might consist of the extracted keywords “car” and “train”, but we would have to interpret this group of

words for ourselves as being related to some sort of topic associated with “transportation”.

The SVD method for LSA works by decomposing the original term-document matrix into a set of three new matrices U, Σ, and VT . The U matrix represents the strength of the relationship between each topic and each term or word, and thus takes the shape *m x r*, with r representing the number of topics. The Sigma matrix is a diagonal matrix of the singular values, each of which is equal to the square root of the eigenvalues of the matrix AT A, and each of which represents the relative importance of each topic. The V matrix represents the strength of the relationship between each document and each topic, and thus takes the shape *r x n*. One use of the SVD for our purposes, is to decompose the original term-document (or in our case sentence-document) matrix into a more compressed form. We achieve this by selecting only the top k topics in our use of the algorithm (provided by scipy.sparse.linalg.svds). This method preserves the most important semantic information from the sentences, while reducing noise and other artifacts of the original vectorization.

After these preprocessing steps we have our reduced *(# sentences) x (# topics)* matrix which can be used by the text summarization algorithm.

3.3 Semantic Volume Maximization

Given a list of sentences (s_1 ... s_n) , with their associated vectorized representations (u_1 ... u_n) , as well as the maximum length of the summary (L), Semantic Volume Maximization returns the set of sentences whose corresponding vectors span a greedily-computed maximal volume of the semantic space. The steps of the algorithm are as follows:

1) Compute the cluster centroid c, the mean vector across all vectorized sentences
2) Find the vector u_p most distant from c
3) Find the vector u_q most distant from u_p
4) Compute b_0, the unit vector corresponding to the vector u_q
5) Add b_0 to the set B, which will store all further basis vectors
6) Add s_q to the final set of sentences S
7) While the total word length of the sentences in S is less than our limit L:
    a) Compute vector U_r most distant from subspace spanned by vectors in B
    b) Add its sentence to S and add its normalized vector representation to B
8) Return the list of sentences S

The way that this algorithm works is by iteratively choosing the vector whose projection onto the existing subspace of bases is maximally distant from the vector itself. The authors of the original paper mention that this method can be thought of as finding a convex hull of the subspace of the sentence vectors, however we have a question to whether this is a proper interpretation. In either case, because of the iterative way in which vectors are added to the set B, Semantic Volume Maximization is a greedy algorithm. This summarization method, with additional options and parameters can be accessed at https://unilever-nlp.herokuapp.com/index.html. An additional parameter can be used to extract noun phrases from the uploaded contents of the excel file.

Our more recent summarization work has involved making improvements to the existing summarization pipeline, through both bug fixes, and additional parameters. The most significant improvement to summarization has been fixing a critical issue in vectorization, by switching out the existing MinMaxScaler with a Normalizer, also from Sklearn. Without this change, instead of standardizing the lengths of the sentence vectors, we were instead standardizing the columns of the data that represent the counts of individual words across each sentence. This fix introduced a noticeable improvement in the results of the algorithm.

The next most important recent change was the inclusion of optional ngram range selectors, to allow for customization of the parameters given to Sklearn's CountVectorizer or TfIdfVectorizer. In theory, this should allow for the inclusion of longer phrases as features to apply to the algorithm, which may be useful for detecting sentences that have not only unique vocabulary, but unique combinations of words as well. For example, take a comparison of the sentences:

"Red, the cat, and the dog"
"The red cat, and the dog"

A vectorization with ngram range 1, would find these sentences to be identical (red, cat, and dog dimensions of 1, excluding dimensions with stopwords). However in a 1,2-gram representation, we would find the following vectorizations:

Red	Cat	Red Cat	Dog
1	1	0	1
1	1	1	1

In this second representation, our algorithm would detect that these sentences are not identical in terms of information, and would thus be more likely to include both sentences in the final summarization. Increasing the bigram count past a value of 2

would further detect these semantic differences, however increasing it too much may cause phrases that are essentially equal in meaning, to be considered different, and thus add redundant information to the final summarization.

The third change we included was a minor improvement to the sentence splitting feature, which divides sentences past a certain length into equally-sized phrases of a specified sublength. Previously this feature only performed a single split. The initial idea for this feature was to provide additional regularization for sentences that were longer, as it was initially proposed that these sentences would be preferentially added to the final list due to their large differences from other vectors. However because of the previously mentioned fix to normalization, this is no longer an issue. It is however interesting to see how well the algorithm performs on sentences that are limited to shorter lengths such as two or three, as in such cases the algorithm seems to approximate a crude form of keywords extraction, by choosing only single words or phrases with larger Tfldf values across the dataset.

Further improvements to the application, not limited to the summarization tasks, include the addition of two test datasets, originating from a study by Tran et al. (2013) which used data from different news sources to test timeline summarization tasks. These datasets provide an example format for our algorithm, and can be used as well to test it. We additionally included further styling changes and error handling functionality into the application for debugging purposes. A small fix was made in the concatenation stage of the pipeline to conditionally introduce periods to the ends of sentences that users may have omitted, in order to aid sentence tokenization and result readability.

4. Keyphrase Extraction

Extracting keyphrases from a long document enables a very quick insight in the content of the text. In our case, the analyst can easily understand the different concerns of customers, without having to go through all survey answers.

Keyphrase extraction is an entire research area in Natural Language Processing and many algorithms exist. The two most commonly used are TextRank [2] and RAKE [1]. Based on the research paper, RAKE achieves better or similar performance on the task itself while being much more computationally efficient. This led us to consider this algorithm first. Furthermore, it is open source (MIT License) and can thus be used at Unilever.

4.1 Description of the original algorithm

It is important to note that the algorithm is unsupervised and can thus be adapted to every language. It matters in our case as it could process surveys conducted among Unilever customers from any country.

Input of the algorithm:

document
list of stop words and phrase delimiters (the, a, from…)
parameters:
minimum length of a word in a keyphrase
minimum frequency for a word in the text
maximum number of words per keyphrase

Output:

List of keyphrases and the associated relevance score

The algorithm starts by parsing the text based on the given parameters and the list of stop words and phrase delimiters. This step results in a set of candidate keyphrases.
Then, a relevance score is computed for each keyphrase, which is the sum of the scores for each word in the keyphrase.

The score of a word in a keyphrase equals the ratio between the degree of the word and the frequency of the word. The degree of a word w is the number of words that appear in keyphrases where w is. The frequency of a word w equals the number of times w appears in the text.
We can then rank all keyphrases based on the relevance scores.

The different steps are illustrated below:

Original text:

Compatibility of systems of linear constraints over the set of natural numbers

Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

Candidate keyphrases:

Compatibility – systems – linear constraints – set – natural numbers – Criteria – compatibility – system – linear Diophantine equations – strict inequations – nonstrict inequations – Upper bounds – components – minimal set – solutions – algorithms – minimal generating sets – solutions – systems – criteria – corresponding algorithms – constructing – minimal supporting set – solving – systems – systems

Score for each word:

	algorithms	bounds	compatibility	components	constraints	constructing	corresponding	criteria	diophantine	equations	generating	inequations	linear	minimal	natural	nonstrict	numbers	set	sets	solving	strict	supporting	system	systems	upper
deg(w)	3	2	2	1	2	1	2	2	3	3	3	4	5	8	2	2	2	6	3	1	2	3	1	4	2
freq(w)	2	1	2	1	1	1	1	2	1	1	1	2	2	3	1	1	1	3	1	1	1	1	1	4	1
deg(w) / freq(w)	1.5	2	1	1	2	1	2	1	3	3	3	2	2.5	2.7	2	2	2	2	3	1	2	3	1	1	2

Candidate keyphrases and the associated scores:

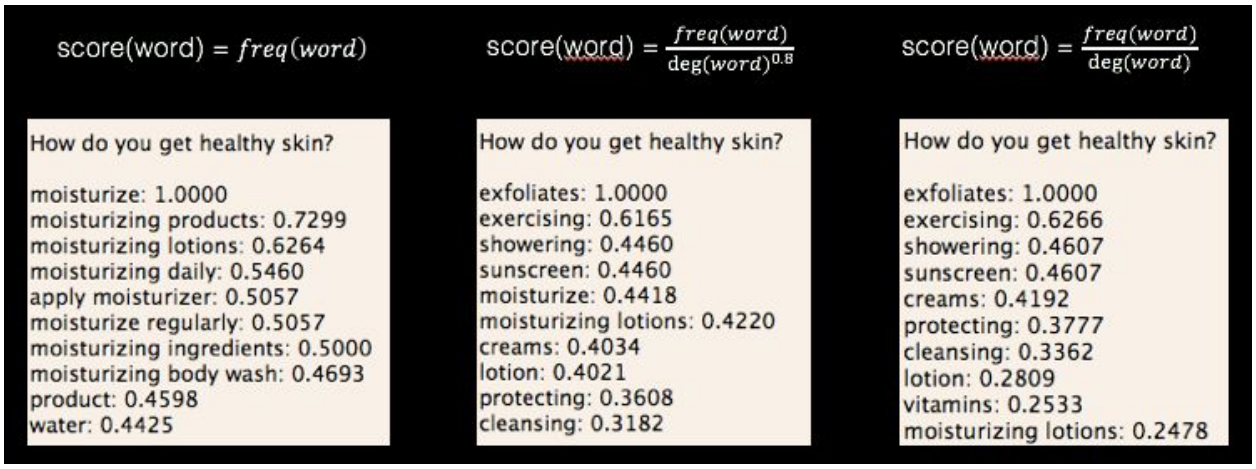
minimal generating sets (8.7), linear diophantine equations (8.5), minimal supporting set (7.7), minimal set (4.7), linear constraints (4.5), natural numbers (4), strict inequations (4), nonstrict inequations (4), upper bounds (4), corresponding algorithms (3.5), set (2), algorithms (1.5), compatibility (1), systems (1), criteria (1), system (1), components (1),constructing (1), solving (1)

4.2 Modifications added to the algorithm

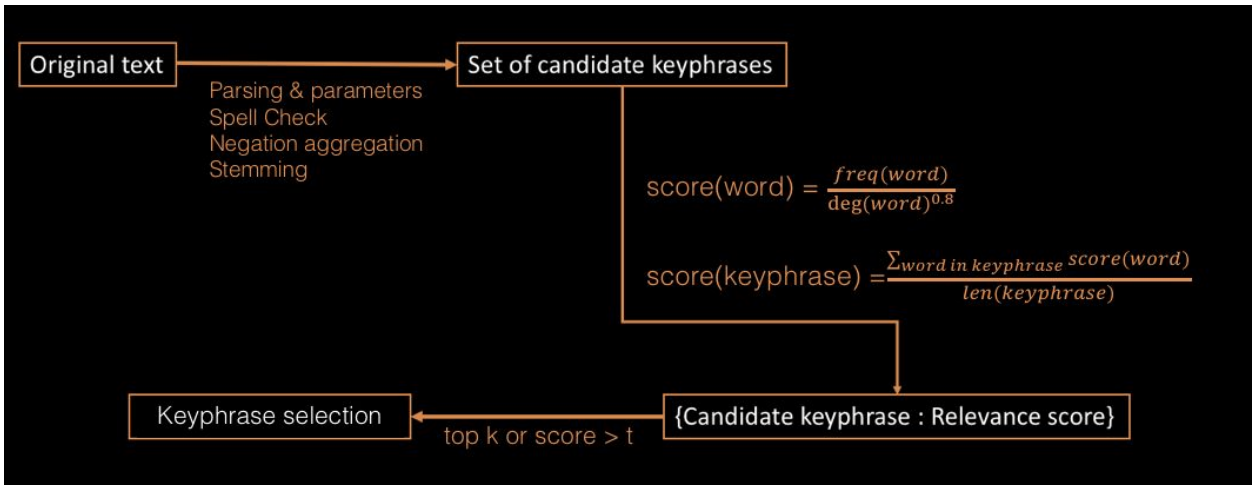
In order to reach a better accuracy and extract keyphrases that make more sense, we had to add the following preprocessing steps.

Spell check
    mosturizd -> moisturized
Use of stemming to group similar keyphrases
    {moisturized, moisturizing} -> moistur
Negations aggregation
    not oily -> no_oily / 2 words -> 1 word
Added parameter
    minimum length of a keyphrase
Score Scaling
    All scores between 0 and 1, using a MinMax Scaler
New scoring function

The current scoring function for a word, score(word)=deg(word)/freq(word) returned rather unsatisfying results, so we tried different combinations, that are summed up below. We manually found that the best trade-off between the frequency and the degree was obtained for the frequency divided by the degree at the power of 0.8.



4.3 Final Pipeline for the algorithm

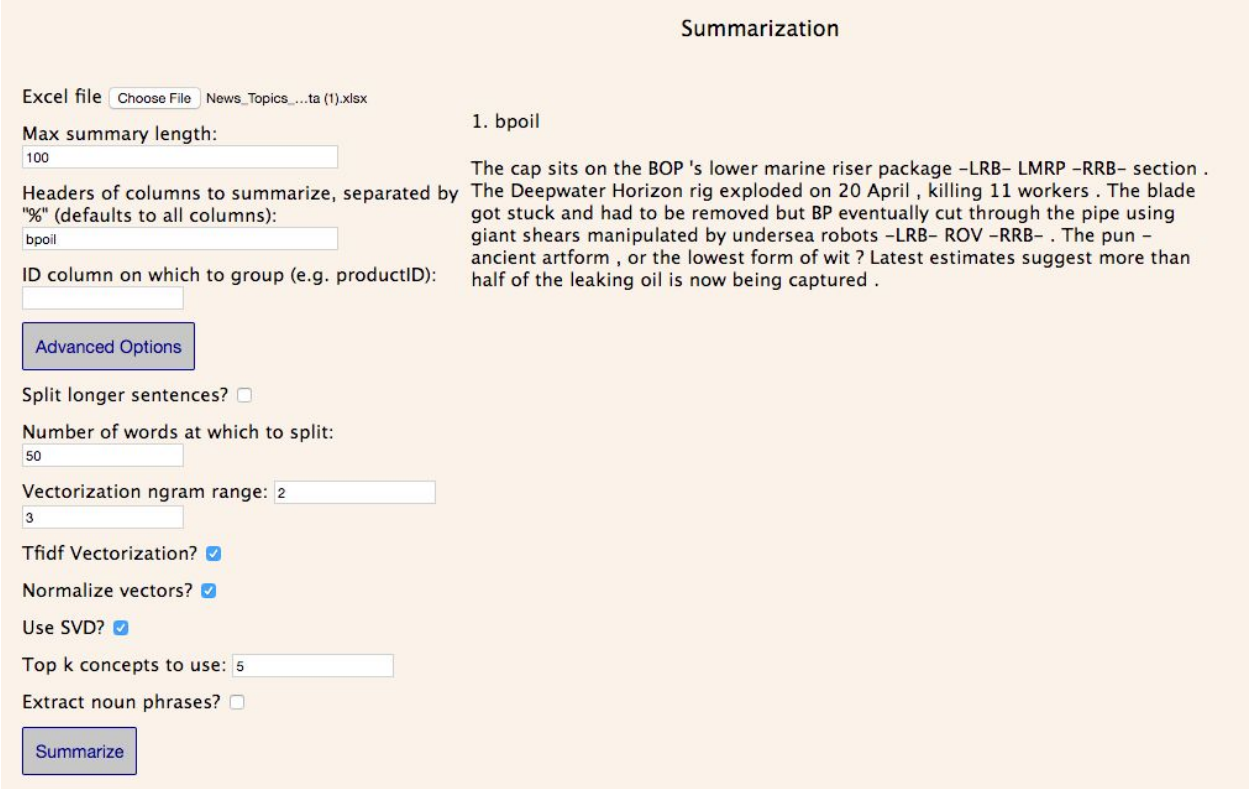


5. User interface and Results

The web interface was built using Python libraries for the algorithms, the web framework Python Flask for the back end, and HTML, CSS and JavaScript for the front end. It is hosted online on heroku (https://unilever-nlp.herokuapp.com/) and the code is on a public github repo (https://github.com/Unilever-NLP/unilever-nlp).

5.1 Summarization on the web interface

Parameter selection



Results

1. What is Healthy Skin?

clean fresh and blemish free. Soft, moisturized, smooth, clean. Moist, soft to touch, no sloughing or cracks. radiant, glowing skin. Good elasticity. Elastic , moisturized, glowing. Not flaky or red or dry. Good color. Healthy skin is smooth, with no breaks in the surface. No flakes. Clear radiant skin. glowing, soft. Using the right products and drinking plenty of water can give you healthy skin. Moisturized, non dry skin, clean skin. Skin that isn't dry or flaky. Clear skin. Even skin tone. glowing moisturized soft skin. Clean clear not dry. normal to dry. No sun damage.

2. How do you know your skin is healthy?

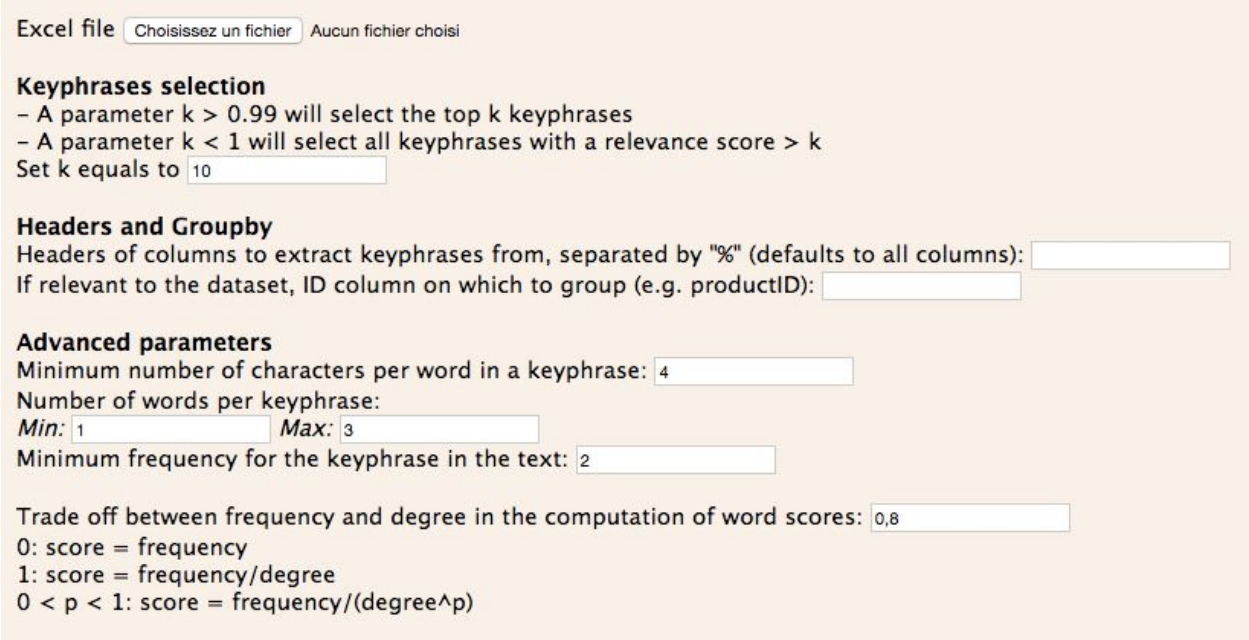
Not sure. Again, not to dry not to oily, a glow??. I exfoliate atleast once a week and I make sure to moisturizer daily. Because I use moisturizing body wash and lotion daily. Soft smooth no rashes no dryness no cracking no shafing. No marks. looks radiant. Even skin tone, clear pores, no flakes. I do what I can to take care of my skin. Soft to touch. smooth, soft. I can tell if it is healthy if it is clean and well moisturized, not flaky. dont know. LOOKS GOOD. glowing look. By a glow. Feels good. Look/feel.

We can see that these results do indeed maximize the semantic space, and after looking through the original documents, we find that most of the pertinent vocabulary is present in the summarization.

5.2 Keyphrase Extraction on the web interface

Parameter selection

All parameters were introduced above or are described in the following screenshot:



Results

The results can be exported to a csv file, if further analysis is required, depending on the task. Following are sample results on the survey data (left) and product reviews (right).

What is Healthy Skin?	B001F51T3Q
healthy skin: 1.0000 / 0.32353	skin silky smooth: 1.0000 / 0.06250
skin no_dry: 0.8844 / 0.02941	sensitive skin version: 1.0000 / 0.06250
healthy skin glows: 0.8248 / 0.01765	oilier skin types: 0.8393 / 0.03125
soft skin: 0.8044 / 0.02941	skin perfectly no_greasy: 0.8181 / 0.03125
glowing skin: 0.7763 / 0.03529	drugstore face lotions: 0.7734 / 0.03125
healthy glow: 0.6980 / 0.03529	cerave moisturizing lotion: 0.7310 / 0.03125
moisturized skin: 0.6612 / 0.02353	face baby soft: 0.6719 / 0.03125
smooth skin: 0.6130 / 0.02353	super soft skin no_flake: 0.6567 / 0.03125
no_dry no_oily: 0.6013 / 0.07059	small sample bottle: 0.6383 / 0.03125
no_oily no_dry: 0.6013 / 0.02353	pores causing acne: 0.6383 / 0.03125

6. Future Developments and Continuation

This project is the first step in the development of a more complex text analysis software and there are many improvements that still can be achieved, by the following Capstone groups.

- Handle larger datasets by porting application to AWS
- Improve the user friendliness of the interface
- Display the original sentence when click on a keyphrase
- Compute the score for a given keyphrase, input by the analyst

There are additionally several features for summarization that have been locally implemented, but have proved difficult to deploy to our remote server due to memory constraints. The first is the subordinate clause extraction feature. This feature makes use of the sentence POS tagging feature of NLTK to divide sentences into their adjacent clauses by extracting sibling ‘S’ tags from the tree. For example:

“This product is well priced and it works as expected.”

Contains two ‘S’ tags, separated by a ‘CC’ tag, and would be extracted to:

“This product is well priced.”
“It works as expected.”

This feature may prove helpful towards eliminating redundancy in the results when subordinate clauses from sentences that are different overall may contain similar content. A second feature which is in this same state of local functionality is misspelled words exclusion. We would like to include this feature in the pipeline to prevent misspellings from being disproportionately overweighted because of their uniqueness.

Two additional features that are not yet implemented include word2vec representations of sentences for higher dimensional comparisons, synonyms aggregation across datasets using synsets, and exploration of an alternative distance metric between sentences and the existing basis subspace using a summation of cosine similarities. We hope to be able to introduce one or more of these features in the final version of the application.

Suggestions for further improvements to the accuracy and usability of the summarization task include:

1) Integrating cosine similarity as a distance metric in place of the distance from the projection onto the basis subspace
2) Separate sentences based on conjunctions and delimiters using Stanford's core NLP
parser and extracting S tags.
3) Utilizing word embeddings for sentence vectorization values

The following group can simply take this report to understand our achievements, then get the code on the github repo (https://github.com/Unilever-NLP/unilever-nlp) and take this project to another level!

7. References

1. Mihalcea, Rada, and Paul Tarau. 2004. TextRank: Bringing order into texts. Association for Computational Linguistics.
2. Rose, Stuart, et al. Automatic keyword extraction from individual documents. *Text Mining* (2010): 1-20
3. Safyan, Joshua, et al. 2016. Keyword Analysis and Automatic Summarization. ENGIE4800 course, Columbia University
4. Steinberger, Josef, and Karel Ježek. 2004. Text summarization and singular value decomposition. International Conference on Advances in Information Systems. Springer Berlin Heidelberg.
5. Tran, Giang Binh, Mohammad Alrifai, and Dat Quoc Nguyen. 2013. Predicting relevant news events for timeline summaries. Proceedings of the 22nd international conference on World Wide Web companion, pages 91–92. International World Wide Web Conferences Steering Committee.
6. Yao, J., Wan, X. & Xiao, J. 2017. Recent Advances in Document Summarization. Knowl Inf Syst. doi:10.1007/s10115-017-1042-4
7. Yogatama, Dani et al. 2015. Extractive Summarization by Maximizing Semantic Volume. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing: 1961-1966. September, 2015.